EMBEDDED LINUX CONFERENCE

@

THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

Porting Linux to a Baseboard Management Controller ASIC: Feedback and Perspectives

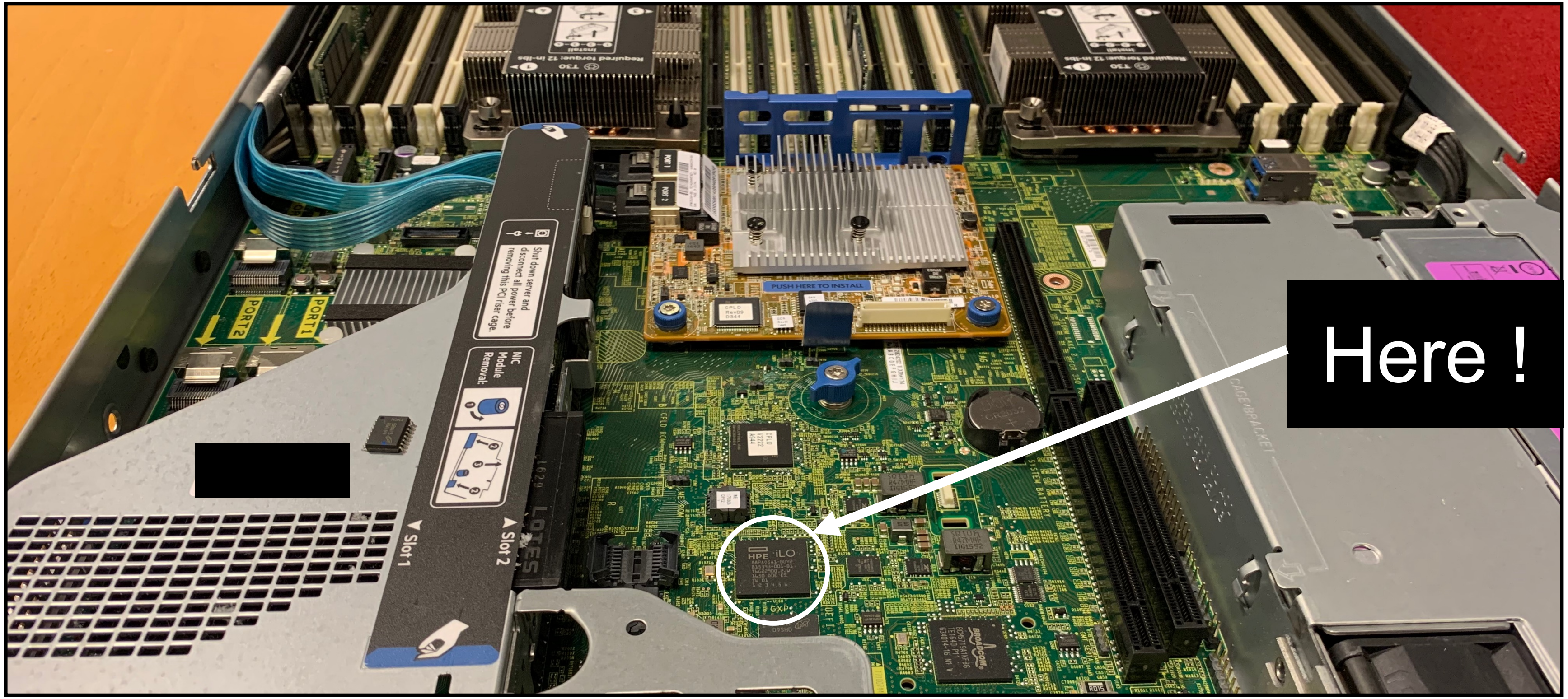
Luis Luciani, HPE, Distinguished Technologist

Jean-Marie Verdun, HPE, Distinguished Technologist

#ossummit

THE LINUX FOUNDATION

Where is the BMC and what does it do?



The iLO firmware - a history based on RTOS(s)

iLO

2002
Embedded Lights-Out Management Processor
HW-based graphical remote console
Digitally signed/secure update of iLO firmware

2003
LDAP integration
Virtual CD

iLO 2

2006
Integrated remote console
High-speed shared network port

2007
AES encryption
Remote console record and playback
Authentication logging

2008
Dynamic power capping

iLO 3

2010
AES HW encryption engines

2011
Kerberos authentication
Two-factor authentication

2012
Full FIPS validation
Secure BIOS update

iLO 4

2012
Active health system
Agentless management
Embedded remote support

2013
SNMP v3

2014
iLO Federation

2015
Full FIPS validation

iLO 5

2017 / 2018
Silicon Root of Trust
Secure Start
Runtime Firmware Verification
Secure Recovery
Server System Restore
Full FIPS Validation
Downgrade Prevention
Security Dashboard
1-Button Secure Erase
Workload Performance Advisor

Ongoing security cipher and vulnerability updates

Why enable Linux on GXP ?

1

Control Plane is integrated into a complex infrastructure
Host is no longer a basic BareMetal infrastructure. Proper setup and monitoring in virtualized environment is a must have feature

2

Linux is well known and understood environment
Easy to access to developer resources, wide range of testing capability and drivers availibility

3

Higher threats, different answers
Security liability vary from end users to end users. Proprietary solution no longer covers the whole spectrum of expectations

4

Common software base between vendors
IT infrastructure evolved from dozens of servers to thousands of server within enterprise world inducing multi vendors sourcing and compatibility challenges within the control plane

5

BMC hardware can now do it
Faster CPU capabilities with better manufacturing process. Good ARM support

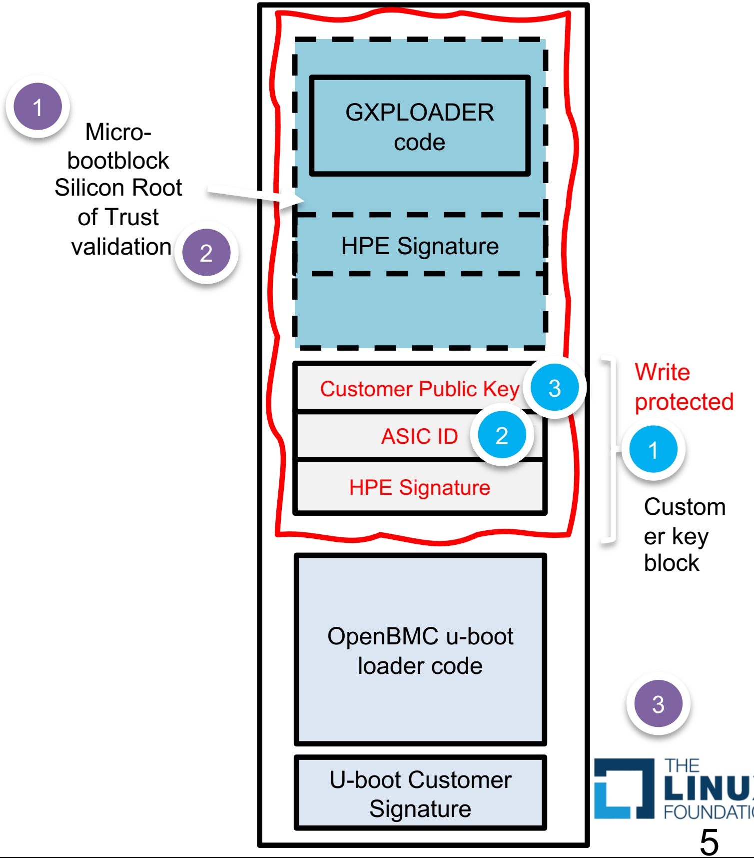
Leveraging GXP security under linux

Transfer of Ownership: Who can sign the firmware binary that runs on the hardware?

- HPE's BMC ASIC ("GXP") designed to run iLO 5 firmware
- Silicon Root of Trust designed to ensure HPE signed firmware
- No flexible "transfer of ownership" in the existing hardware

Proof of Concept Solution: Chain of Trust

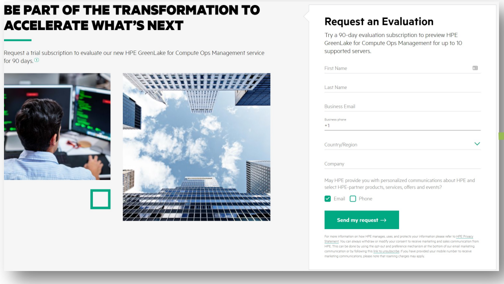
- HPE GXPLOADER Binary- HPE signed and can run on ASIC
- Requires Customer Key Block
  - HPE Signed
  - ASIC locked to a unique ASIC ID value
  - Contains customer public key for U-BOOT validation



Upcoming transfer of ownership

HPE.com site

End user must be able to build and sign their own OpenBMC firmware



End user request comes to the PMO
After approval, end user sends HPE their public key

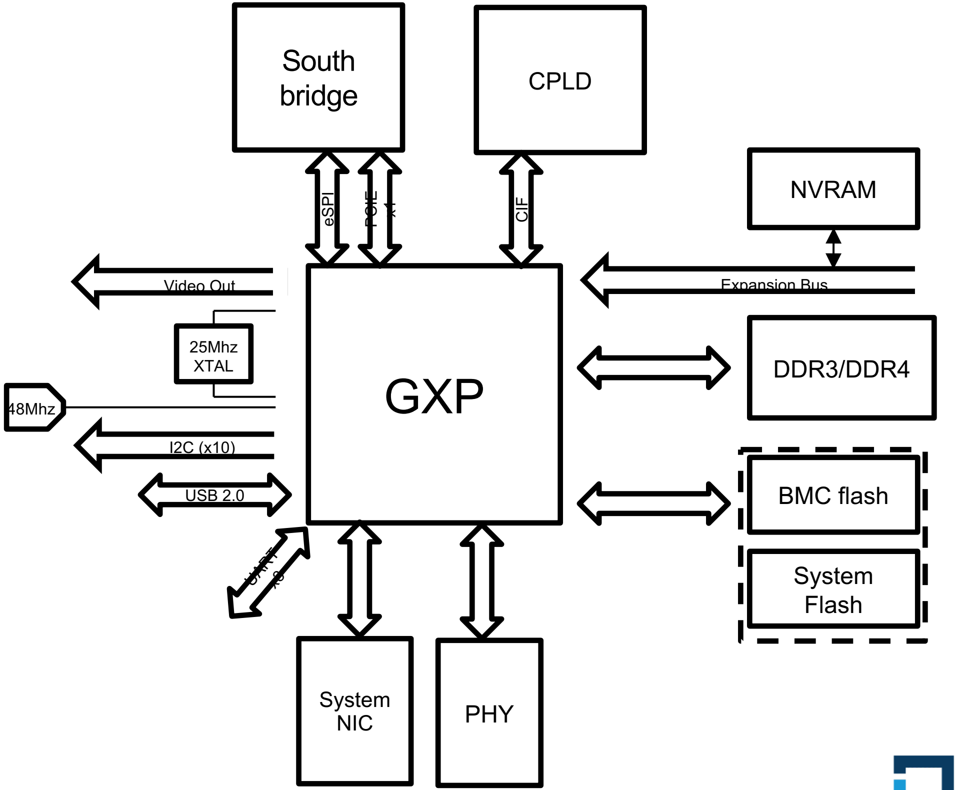
HPE sends back an image for a USB key
End user adds their:

- iLO credentials
- Signed OpenBMC image

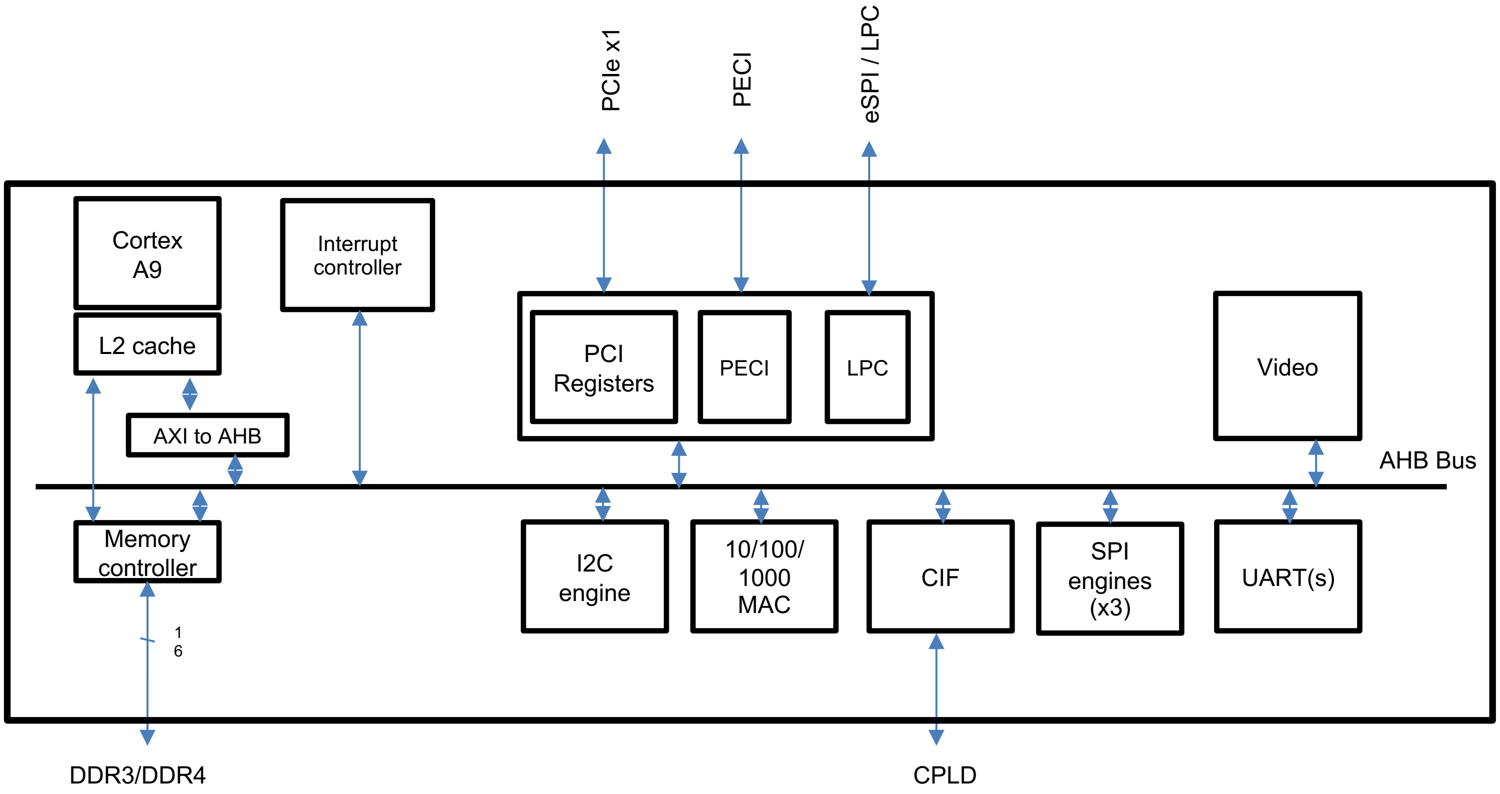
End user inserts the USB key and powers on
Automatically:

1. Ownership is transferred
2. ROM is installed
3. OpenBMC is installed
4. Light turns blue

GXP BMC programmable interfaces



GXP Internals



What are these buses used for ?

PECI (Platform Environment Control Interface)

- **Control Thermal management Reporting**
- **Single wire bus**
- **Digital Thermal sensing (ΔT)**

CIF (CPLD InterFace)

- **Proprietary Bus**
- **Self training, x1 to x8 lanes, up to 266Mhz clock**
- **Packet Protocol**
- **Focused on GPIO, FAN status and PWM**

CPLD (Complex Programmable Logic Device)

GPIO

- x2 64 bits General purpose GPIO tunnelled through CIF
- Mainly Host driven (PGOOD etc…)
- Power control sequencing
- Host status

Thermal management

- x8 PWM for fans
- Programmable duty cycles (256 steps)
- Thermal protection watchdog timer
- Fan Fault

THE LINUX FOUNDATION

GXP Main address spaces

Core Registers (GXP internal memory mapped register)

Access 8-bit,16-bit, 32-bit ok	Address Range Start	Ending
Core	$c000_0000	01bf
Global Display	$c000_01c0	01ff
SPI	$c000_0200	03dd
DVR	$c000_0400	046f
Thumbnail Control	$c000_0500	05ff
Fan control	$c000_0c00	0dff
CIF	$c000_1000	11ff
I2C engines (x10 with $100 stepping)	$c000_2000	3fff
Primary Ethernet	$c000_4000	41ff
Secondary Ethernet	$c000_5000	51ff
SPI ROM Ext Data #0	$c000_c000	dfff
SPI ROM Ext Data #1	$c000_e000	ffff
USB HC UTMI PHY Registers	$c001_1000	10ff
USB HC EHCI	$cefe_0000	00ff
USB HC OHCI	$cefe_0100	02ff
ARM VIC	$ceff_0000	0fff

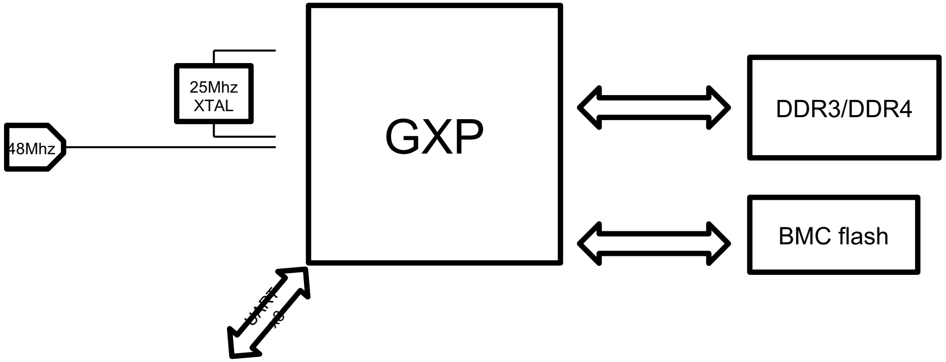
Host Registers

Access 8-bit,16-bit, 32-bit ok	Address Range Start	Ending
SysSupport Configuration	$800f_0000	0fff
SysSupport Configuration	$8000_0000	07ff
SMI Services	$800e_f000	f0ff
MgtSupport Configuration	$802f_0000	0fff
MgtSupport	$8020_0000	00ff
vEHCI PCI Configuration	$804f_0000	0fff
vEHCI Host Controller Runtime Memory Mapped	$8040_0000	07ff
vDevice Global Registers	$8040_0800	08ff
vEHCI Virtual Device Registers	$8040_1000	1fff
PSP PCI Configuration	$805f_0000	0fff
FN5 UART Device	$8050_0000	00ff
SROM Configuration	$80fc_0000	00ff
SROM Device	$f000_0000	$f7ff_ffff
System UART A Device	$80fd_0200	02ff
System UART B Device	$80fd_0300	03ff
System UART C Device	$80fd_0800	08ff
Legacy KCS Device	$80fd_0400	04ff

Expansion device space (CPLD)

Access 8-bit,16-bit, 32-bit ok	Address Range Start	Ending
CIF X-Reg area	$d100_0000	$d1df_ffff
CIF Address space	$d1e0_0000	$d1ff_ffff

Linux port where do we stand ?



- PoC of all the drivers are available
- Ongoing upstreaming process
  - 5.20 window
    - Clock
    - DTS
    - UART
    - Watchdog
- U-boot upstreaming process started
- GXP-Loader is done and published
  - https://github.com/HewlettPackard/gxp-bootblock

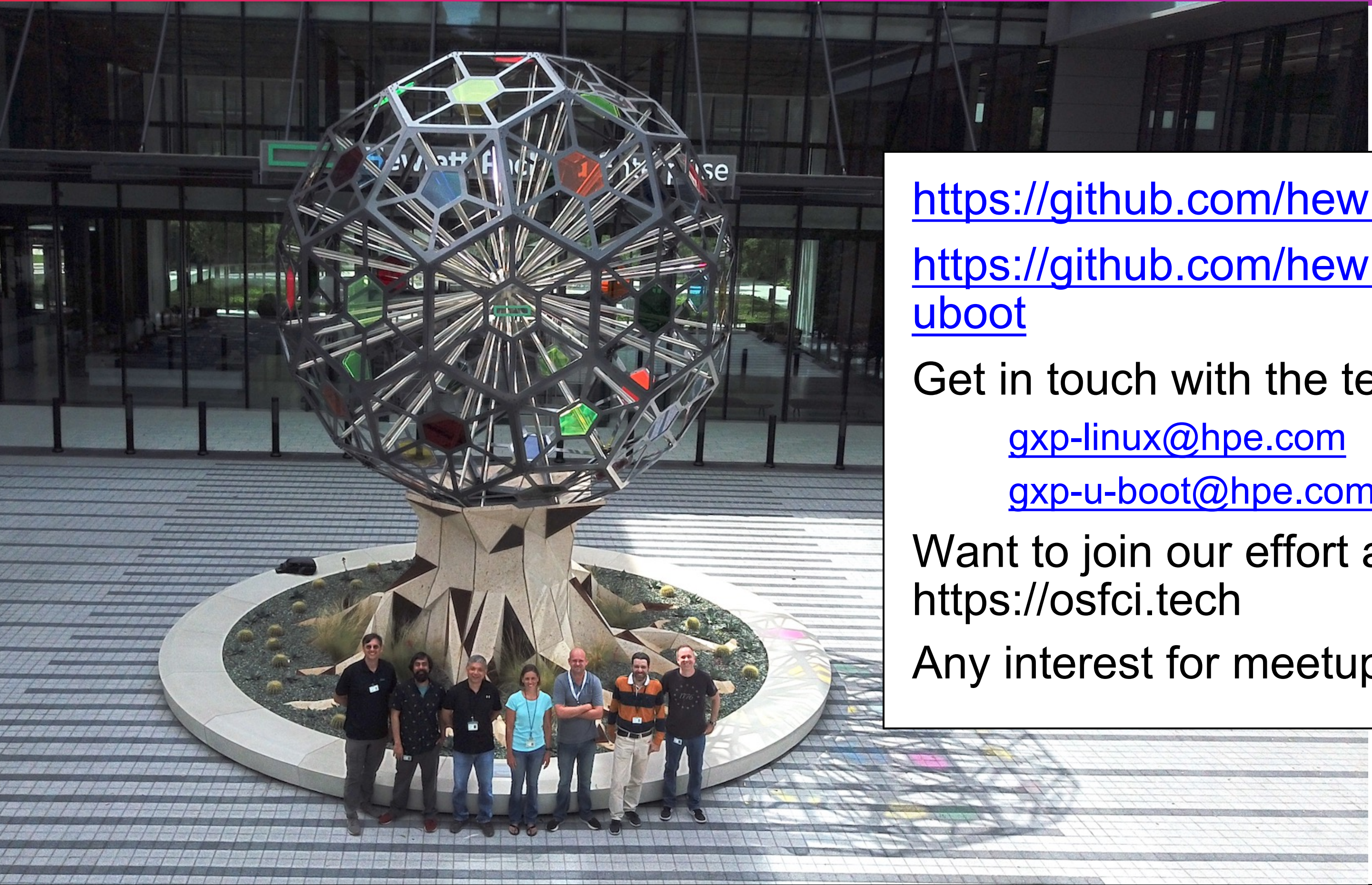
From PoC to upstream

- Challenges to:
  - Understand from where to start
    - Yaml documentation of the DTS node
    - DTS styling
    - Initial drivers
    - What is a minimum basic acceptable state for a new SoC into the kernel ?
  - Adopt relevant code styling
- But the community has proven to be extremely patient with us, we really appreciated this.
- ARM linux kernel support is good but complex coming from SoC fragmentation and lack for standard
  - Defaulting to a standard defconfig ended to integrate a new ARM Errata (unexpected on an aging Cortex A9)
- BMC are new beast with specifics which require standardization that can happen only through multiple stepping process
- Drivers need access to multiple memory regions which could lead to weird dts entry

What is next ?

- Infrastructure to support host start
  - Network driver enablement
  - SPI setup
    - SPI Driver to read and copy initial ROM content within vROM
    - CPLD configuration for virtual ROM startup
  - GPIO setup for
    - Basic power button control
    - Power sequencing control
    - Events driven interrupt (Fan / hotplug)
  - Fan controller
  - Host vUART
  - Video encoder
  - Virtual USB hub (Keyboard/Mouse/VirtualNIC)
  - KCS driver
  - CHIF high speed tunneling for RAS logging

Want to help ?



https://github.com/hewlettpackard/gxp-linux

https://github.com/hewlettpackard/gxp-uboot

Get in touch with the team:

- gxp-linux@hpe.com
- gxp-u-boot@hpe.com

Want to join our effort and develop ?
https://osfci.tech

Any interest for meetups ?

EMBEDDED
LINUX
CONFERENCE

@

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA